

• Popular Computing

The world's only magazine devoted to the art of computing.

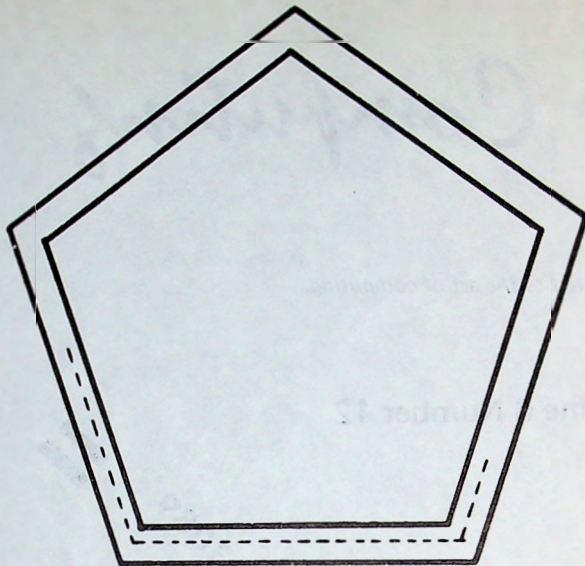
December 1978 Volume 6 Number 12

80371212674509332988345845936510711253089

2

...112122111122212112212111212111211121111212122112





The outer regular pentagon in the Figure has an area of 10,000 square units, and the inner pentagon has an area 100 square units less, or 9900 square units. The length of the strip between them is taken to be the perimeter of the dotted pentagon, which lies half way between the other two.

One hundred such strips are removed from the original large pentagon; the second one has an area of 99 square units; the third one an area of 98 square units; and so on, to the 100th one, with an area of one square unit.

What is the total length of the 100 strips?

PROBLEM 251

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is \$20.50 per year, or \$17.50 if remittance accompanies the order. For Canada and Mexico, add \$1.50 per year. For all other countries, add \$3.50 per year. Back issues \$2.50 each. Copyright 1978 by POPULAR COMPUTING.

Publisher: **Audrey Gruenberger**
 Editor: **Fred Gruenberger**
 Associate Editors: **David Babcock**
 Irwin Greenwald
 Patrick Hall
 Contributing Editors: **Richard Andree**
 William C. McGee
 Thomas R. Parkin
 Edward Ryan
 Art Director: **John G. Scott**
 Business Manager: **Ben Moore**

What Should We Compute? -II

"I do not ... question the ability of anyone to find problems that seem to require high precision, but I do wonder about whether the problem is worth solving in the form given."

--R. W. Hamming

[The material in this essay is adapted from
the RAND Paper P-2786 dated September 1963]

The eighth article in this series (in issue No. 28, July 1975) was titled "What Should be Computed?" In that article, an old puzzle problem (the Diophantine problem of the bank check) was used to highlight the six criteria for intelligent use of a computer. Those criteria are reviewed here.

1. The problem solution must be useful to someone.
This is obvious by definition; few people would condone working on useless problems. The tricky part comes when you ask: Useful to whom? While one is learning computing (a process that may occupy 30 years or more), any problem that improves one's skill and knowledge of computing is ipso facto useful.

Usefulness has another aspect, too. It would be nice if each of us could devote all our efforts to that which is useful to us. Sometimes, though, we are forced to work on some dull--or utterly useless--project that is only useful to someone else. Everyone who does any considerable amount of computing has met the question "Why in the world would anyone want to compute that?" Well, why, indeed? It is difficult to pinpoint just what usefulness means. One can contemplate the people who collect stamps, or buttons, or 1953 Chevrolets, or model trains, or....

2. The problem must be defined. It would be silly to try to solve a problem (with or without a computer) before knowing just what the problem is. When the solution involves a computer, this criterion says that a precise statement of the inputs and outputs is needed.

3. We must have a method of solution (with or without the computer). If we know precisely what the problem is (and we should continually emphasize "precisely") then we need some sort of algorithm for our proposed solution. Note that we say a method of solution. There are usually many ways to solve a problem. It is with this criterion that we have the first distinction between a computer solution to a problem and any other solution. We are privileged to keep in mind that a computer is fast; we may capitalize on this fact to seek, as a first cut, a brute-force (or crowbar) method. Often a problem can be solved by exhaustion, by which we attempt to try every possibility. This is frequently acceptable as a method to use with a computer.

4. The proposed solution must fit the available machine in space and time. There are three elements to this criterion. The first is the available machine; usually this limits us to one machine, and moreover one that is selected for us. A solution that might be perfect on some other machine is useless if we can't get at that other machine. "Space" refers to the amount of real or virtual storage. "Time" refers to the execution time of the program. The solution must be available while it is still applicable.

Brute-force methods may kill us here. Even in a classroom atmosphere (perhaps especially in that atmosphere) computer time is not unlimited, and is almost never free. In a well-run computing course, the student does many exercises. He should also do at least one problem. The distinction is this: an exercise relates to a specific technique, and the approach is usually spelled out in detail. A problem, on the other hand, will involve a broad goal, using many techniques, and with very little spelled out. Part of the student's task is to choose a suitable problem; that is, one suited to his own capabilities and to the machine time available. Hopefully, it should also be a good computer problem. The instructor must prevent the student from going to either extreme (that is, too trivial a problem, or one that the student cannot hope to complete) and should also guide him to a problem that is, according to the criteria we are developing here, a problem worth putting on a computer. All such situations can be covered by furnishing a stock collection of problems; this would satisfy 90% of any class. But the best situation of all is the one where the bright student selects a problem all his own; perhaps one never tackled before.

5. There should be repetition in the proposed solution--the more the better. We can make the statement categorically: a good computing problem has a large element of repetition in the solution. This rule has many exceptions, of course. The use of a computer for true one-shot calculations is fairly common, but that doesn't mean that one-shot solutions (and particularly straight-line formula evaluations) constitute intelligent use of a computer. The thing that a computer does best is to repeat what it just did, and a good problem will capitalize on this capability. Notice that the repetition involved here is in the solution, not in the problem.

6. The proposed solution should be cost/effective; what is sometimes called a low price/performance ratio. The question is, have alternative means of solving the problem been considered? Every tool of information processing that existed before computers is still in existence and available, such as card files, punched card equipment, desk calculators, pencil and paper, graphical methods, and mathematical methods. It is interesting to recall how far the world had advanced up to 1953 (the year that computers began to have an impact) with no more than desk calculators and punched card gear as mechanical aids. We had electronic television, jet engines, and big atomic bombs. Many complex problems had already been solved by the time the first computer went on the air.

The six criteria listed above may be made more vivid by considering the following problem.

Figure 1 shows a short table of powers of 2. In 1949, R. J. Walker noticed that some powers of 2 have a string of the digits 1 and 2 at their low order end. In Figure 1, the 89th power has had its 4 low order digits made bolder to illustrate the matter.

Walker conjectured that this phenomenon is general, and submitted it as a problem to the American Mathematical Monthly, where it appeared in the Advanced Problems section. Specifically, his conjecture was this: for any value of R , is there a value of X such that the X th power of 2 will have R digits that are all 1's and/or 2's at the low order end?

In May 1950, E. P. Starke's proof was published. Most people would have great difficulty in following that proof; it is in the condensed and elegant form that mathematicians cherish, and it assumes a great deal of other knowledge, such as the properties of the Euler ϕ -function. Nevertheless, we can assume that the conjecture has been proved. Moreover, it implies that there are infinitely many of each case. Thus, we are assured that for $R = 138$, say, there is some power of 2 for which the low order 138 digits are all either 1's or 2's in some order. Since there are infinitely many such powers of 2, we would like to find the first appearance. From our Figure 1 we have the following table (Figure 2).

The proof of the conjecture assures us that this table can be extended indefinitely, and in doing that, we have a problem that fits the six criteria superbly. Let's not be greedy; let's develop the table up to $R = 30$.

R	X
1	1
2	9
3	89
4	89

Figure 2.

To the casual observer, the production of this table is not something useful. However, our immediate goal is to make the criteria vivid, and for that purpose we have turned a "useless" problem into something highly useful.

001	2	053	9007199254740992
002	4	054	18014398509481984
003	8	055	36028797018963968
004	16	056	72057594037927936
005	32	057	144115188075855872
006	64	058	288230376151711744
007	128	059	576460752303423488
008	256	060	1152921504606846976
009	512	061	2305843009213693952
010	1024	062	4611686018427387904
011	2048	063	9223372036854775808
012	4096	064	18446744073709551616
013	8192	065	36893488147419103232
014	16384	066	73786976294838206464
015	32768	067	147573952589676412928
016	65536	068	295147905179352825856
017	131072	069	590295810358705651712
018	262144	070	1180591620717411303424
019	524288	071	2361183241434822606848
020	1048576	072	4722366482869645213696
021	2097152	073	9444732965739290427392
022	4194304	074	18889465931478580854784
023	8388608	075	37778931862957161709568
024	16777216	076	75557863725914323419136
025	33554432	077	151115727451828646838272
026	67108864	078	302231454903657293676544
027	134217728	079	604462909807314587353088
028	268435456	080	1208925819614629174706176
029	536870912	081	2417851639229258349412352
030	1073741824	082	4835703278458516698824704
031	2147483648	083	9671406556917033397649408
032	4294967296	084	19342813113834066795298816
033	8589934592	085	38685626227668133590597632
034	17179869184	086	77371252455336267181195264
035	34359738368	087	154742504910672534362390528
036	68719476736	088	309485009821345068724781056
037	137438953472	089	618970019642690137449562112
038	274877906944	090	1237940039285380274899124224
039	549755813888	091	2475880078570760549798248448
040	1099511627776	092	4951760157141521099596496896
041	2199023255552	093	9903520314283042199192993792
042	4398046511104	094	19807040628566084398385987584
043	8796093022208	095	39614081257132168796771975168
044	17592186044416	096	79228162514264337593543950336
045	35184372088832	097	158456325028528675187087900672
046	70368744177664	098	316912650057057350374175801344
047	140737488355328	099	633825300114114700748351602688
048	281474976710656	100	267650600228229401496703205376
049	562949953421312	101	535301200456458802993406410752
050	1125899906842624	102	070602400912917605986812821504
051	2251799813685248	103	141204801825835211973625643008
052	4503599627370496	104	282409603651670423947251286016

Figure 1.

The first 104 powers of 2. The functional values are exact through the 99th power, and are truncated to 30 digits for the last 5 powers.

The problem is defined if it is clear what we seek. The values in Figure 2 were obtained from Figure 1 by eye. We are now at the point where we seek the smallest power of 2 for which the five low order digits are all 1's and/or 2's. When we find that result, we will then proceed to case 6.

Since we are seeking, for each case, the smallest power--that is, the first appearance--having been assured that each case occurs infinitely many times--the problem is serial in nature. We will not be able to solve it any better with two computers, or with all the computers in the world at our disposal. The problem is also intrinsically decimal in nature; we seek a pattern in the decimal representation of the powers of 2. If we use a binary machine to solve this problem, then we will have to pull tricks to simulate decimal action. (There are probably no decimal computers still in operation today; in 1963 nearly half of all the machines in the world were decimal.)

We need a method of solution. A method leaps to mind immediately; namely, to extend the table of Figure 1 and search each value of the function for R 1's and 2's at the low order end. The flowchart of Figure 3 applies. At Reference 1 we set up the initial conditions, which represent the situation we had in Figure 2. At Reference 2, we develop one new line of the table of powers of 2. The argument increases by one; the functional value is formed by doubling. At Reference 3 we arrange to search the new functional value for the digits we want; if we do not find them, we return to Reference 2 to generate another line of the table. If we do find them, we print R and X and, for good measure, F. R is then incremented by one. We now jump back to the search.

We have observed, in Figure 2, that case 3 did not occur in isolation; case 4 came with it. In other words, although there are powers of 2 that have just three low order 1's and 2's:

for example, 2^{389} ends with ...818112,

the first appearance of case 3 brings also case 4. We are observing the Tiger principle, so well stated by Forman Acton (in Numerical Methods That Work):

"This example is horrifying indeed. For if we have actually seen one tiger, is not the jungle immediately filled with tigers, and who knows where the next one lurks?"

The repetition of 89 in Figure 2 was fortunate; we observed the tiger right away and took appropriate action. After incrementing R, we go to Reference 3, not Reference 2.

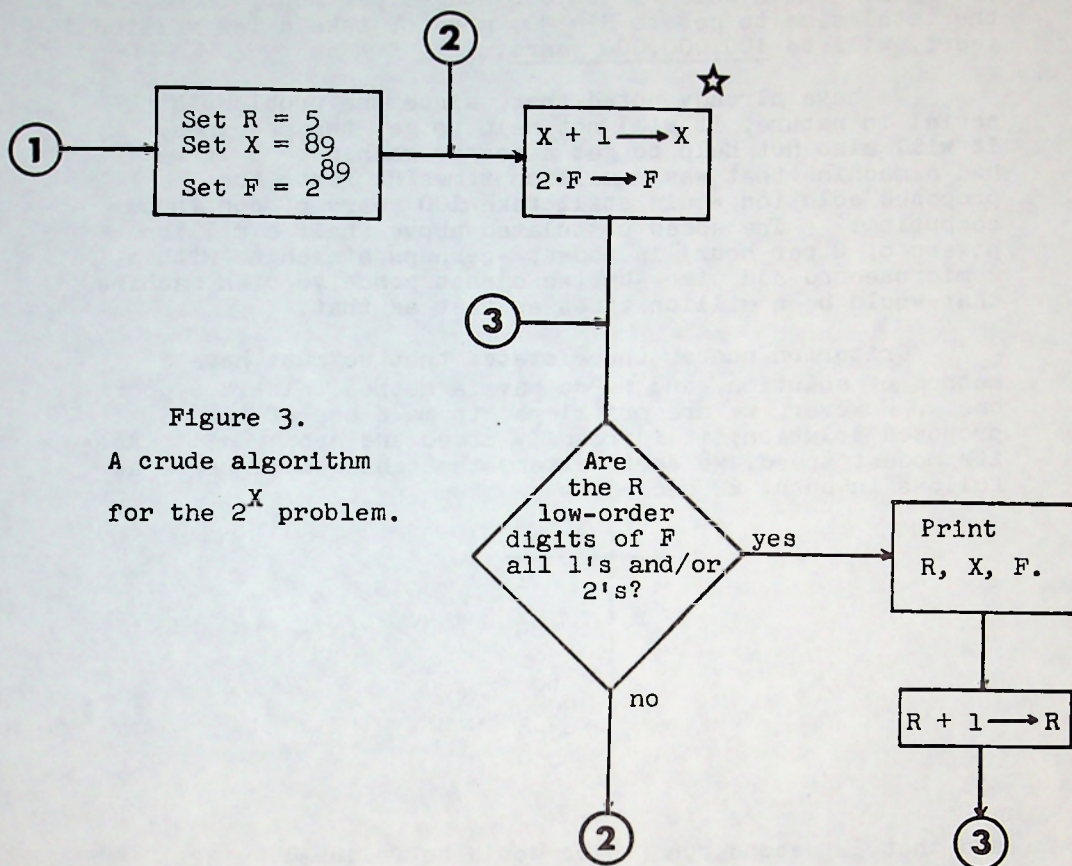


Figure 3.
A crude algorithm
for the 2^X problem.

Our proposed solution must fit the available machine. First, the instructions and data must fit in storage. The instructions for the scheme of Figure 3 might run to 500 or so, considering that we will have to do high-precision decimal arithmetic. For data storage, we will need 2 decimal digits for R (we defined our problem to go to $R = 30$), and 30 decimal digits for F (we can truncate F just the way it was done in Figure 1). The functional values get very large very quickly, but we do not want to examine more than 30 digits. Since multiplication works from right to left, if we retain the low order 30 digits at each stage, we will be correct.

We must also reserve data space for X, but that is unknown. We can reserve, say, 100 digits for X, but with appropriate alarms built into our program so that in case X does try to exceed 100 digits, the program will immediately halt.

Our proposed solution must also fit the available machine in time. If we can implement the flowchart logic of Figure 3 at a rate of 500,000 stages per hour, then the total time to get to $R = 30$, give or take a few million years, will be 100,000,000 years.

We have already noted that, since the problem is serial in nature, it will not help to get two machines. It will also not help to get a faster machine. If we had a machine that was a million times as fast, the proposed solution would still take 100 years of continuous computing. The speed postulated above (half a million powers of 2 per hour) is modest--perhaps a machine with a 2 microsecond add time--but we cannot conceive of a machine that would be a million times as fast as that.

Criterion number three stated that we must have a method of solution, and we do have a method, albeit a poor one. However, we are privileged to make use of our proposed solution; it is readily coded and debugged. At its modest speed, we could extend the table of Figure 2 as follows in about 22 seconds:

R	X
1	1
2	9
3	89
4	89
5	589
6	3089
7	3089
8	3089

Figure 4.

and that 22 second run (which would be followed by 32 minutes of computing with no output, if it were allowed to continue), while inefficient, certainly would suggest a way to improve things. That string of 89's is hardly a coincidence.

Go back to Figure 1. If we look at the units digit of the powers, we know a priori that they must repeat within a cycle of ten, since there are only ten decimal digits and we are constantly multiplying by 2. Actually, they repeat on a cycle of four: 2, 4, 8, 6, 2, 4, 8, 6, ... endlessly. If the numbers we seek must end in 1 or 2, then only every fourth number we have been generating is even eligible. If we search every power, then three out of four searches are automatically wasted (and the bulk of the computer time we are talking about is spent in the searches, not in generating the powers). Not only do we not need to search, we need not even generate the intervening powers. We could advance in one step from, say, the fifth power to the ninth power with one multiplication (by 16, the 4th power of 2) rather than with four doublings.

Just this simple change speeds up the solution by a factor of four. Our 100,000,000 years is now 25,000,000 years; a tiny amount of brain work has saved 75,000,000 years of continuous computation.

We can extend this idea. The low order two digits must repeat within 100 lines, and actually repeat on a cycle of 20. Within those 20, we have already arranged to examine only every fourth line, so we are now considering the cycle that goes 32, 12, 92, 72, 52, 32, 12,...and so on. Again, only one term of this sequence satisfies the conditions of the problem; namely, the term 12. If we can arrange to get into synchronization properly with the powers of 2, we can jump 20 steps at a time by multiplying by the 20th power of 2. We are now down to 5,000,000 years.

Let us now state two theorems that apply. First, the principle hinted at above is quite general. The low order digit repeats with a cycle of four, which is obvious on inspection. The two low order digits repeat with a cycle of 20, which can also be observed from Figure 1. The three low order digits have a cycle of 100, and each additional digit from there on increases the cycle length by a factor of five (the proof of this can be found in the June 1951 issue of the American Mathematical Monthly).

It can also be proved that the terminal sequence of digits consisting entirely of ones and twos is unique. That is, when we find (for $X = 89$) that the last four digits are 2112, then we know that for a longer set of satisfactory terminal digits, the last four will be 2112. Similar statements can be made for the last R digits, for any R .

The material that is being presented in this essay is the end result of about a year's work by a team of researchers--the improvements in the algorithm and the incorporation of "obvious" shortcuts did not occur suddenly. The two theorems stated above, for example, represent steps in the process that took several months, during which many computer runs were made with simple modifications of the flowchart of Figure 3.

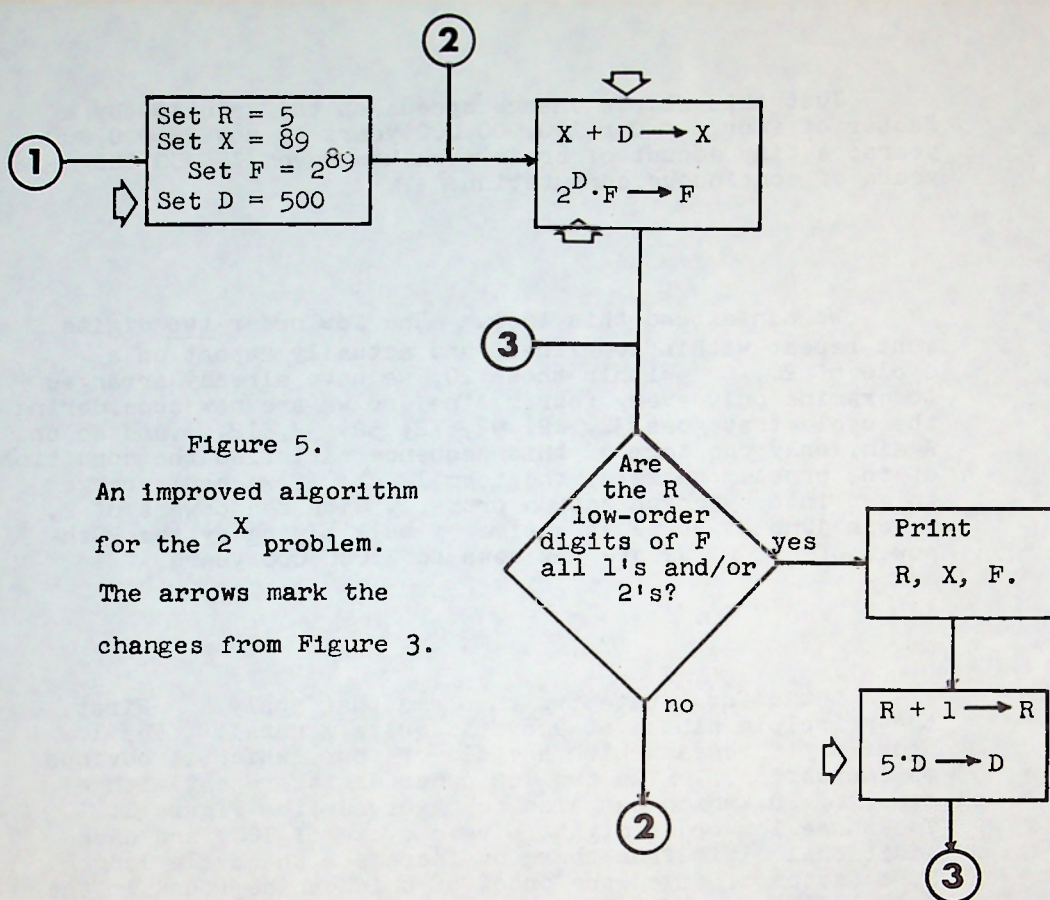


Figure 5.
An improved algorithm
for the 2^X problem.
The arrows mark the
changes from Figure 3.

So what we have is this. The powers of 2 can be explored by ones until we find the X for $R = 1$ (this occurs immediately, with $X = 1$). We have now locked into the solution as far as the units digit is concerned, and we can now proceed four steps at a time; that is, the box marked with a star in Figure 3 could be replaced with the following:

$$X + 4 \rightarrow X$$

$$2^4 \cdot F \rightarrow F$$

We will thus examine only those powers ending with the digit 2. The multiplication by 16 will not increase the computation time significantly, but we will by-pass 3 out of 4 searches.

When we next achieve success (at $R = 2$, $X = 9$, which we reach in two steps), we can start jumping by 20's; that is, we now have:

$$X + 20 \rightarrow X$$

$$2^{20} \cdot F \rightarrow F$$

At each new level, the next digit we seek (called the critical digit) maintains its parity; that is, if odd, it stays odd and if even, it stays even. We are thus assured, at each level, of success within at most 5 steps.

In other words, the algorithm expressed in Figure 3 is crude indeed. The logic of Figure 5 is much better; the changes from Figure 3 have been marked with arrows. If we begin, as shown, at the point where we are seeking the result for $R = 5$, we are already in position to jump by 500's, and the next result ($R = 5$, $X = 589$) occurs in one step instead of 500 steps. The jump rate, D , is then changed to 2500, and the result for $R = 6$ ($X = 3089$) again occurs in just one step.

We must be able, of course, to produce any given power of 2 on demand. This is readily accomplished by calculating a small table of powers of powers of 2, as shown in Figure 6. Every number involved in this problem can be expressed modulo ten to the 30th power; that is, retaining only the low order 30 digits, since we defined the problem that way.

B	$Y=2^B$	2^Y
0	1	2
1	2	4
2	4	16
3	8	256
4	16	65536
5	32	4294967296
6	64	3709551616
7	128	1768211456
8	256	3129639936
9	512	9006084096
10	1024	4224137216
11	2048	9596230656
12	4096	3154190336

Figure 6.

A table of powers of powers of 2,
reduced modulo 10^{10} .

The sample table shown in Figure 6 shows the low order ten digits, to illustrate the principle involved. Thus, for example, to calculate the 89th power of 2, we express the exponent 89 in binary:

1 0 1 1 0 0 1
(7) (6) (5) (4) (3) (2) (1) -- position number

and extract the values from lines 1, 4, 5, and 7 from the table, and multiply them together (reducing every product to the low order 30 digits).

The problem was defined to go to $R = 30$ simply because that led to the value of 100,000,000 years. With the improved algorithm of Figure 5, the calculation to $R = 30$ on the same machine takes 20 minutes, which is quite a dramatic improvement. The last time this problem was run, it went to $R = 60$. Some of the last results from that run are shown in Figure 7. Figure 7B, below, shows some of the earlier results.

R	X
1	1
2	9
3	89
4	89
5	589
6	3089
7	3089
8	3089
9	315589
10	315589
11	8128089
12	164378089
13	945628089
14	1922190589
15	11687815589
16	109344065589
17	231414378089
18	1452117503089
19	4503875315589
20	65539031565589

Figure 7B.

We have practically demolished this problem. If there were any point to extending the results beyond $R = 60$, the way is clear.

That was Problem A. We have also Problem B. Go back once more to Figure 1. It was observed that the 10th power of 2 is the first to exhibit (in the decimal expansion) the digit zero, and the 53rd power is the first to exhibit two embedded contiguous zeros. Problem B is this: For any R , is there an X such that the X th power of 2 contains R embedded contiguous zeros?

55 213110296573030769759639500353484690589
 89580211112221211221212111212112211111211111212122112

56 435154901498062077844365833971648753089
 58892211112221211221212111212112211111211111212122112

57 2655600950748375158691629170153289378089
 52012211112221211221212111212112211111211111212122112

58 24860061443251505967164262531969695628089
 83212211112221211221212111212112211111211111212122112

59 80371212674509332988345845936510711253089
 11212211112221211221212111212112211111211111212122112

60 80371212674509332988345845936510711253089
 11212211112221211221212111212112211111211111212122112

This is R



This is the exponent on 2

...and these are the 60
 low-order digits of
 that power of 2.

Figure 7. Final results of the 2^x problem.

Powers of 2

4327 [1949, 39]. *Proposed by R. J. Walker, Cornell University*

In attempting to solve the preceding problem* we tried to show that for some positive integer s no power of 2 had its last s digits all powers of 2. Show that this attempt had to fail; in particular, show that for each s there exist powers of 2 each of whose last s digits is either 1 or 2.

Solution by E. P. Starke, Rutgers University. The proof proceeds by induction. Noting that $2^5 = 32$ and $2^9 = 512$, let us suppose that 2^n is a number whose last s digits are exclusively 1's and 2's, so that

$$2^n \equiv z \pmod{10^s},$$

where the s -digit number z is composed entirely of 1's and 2's. From the properties of the Euler ϕ -function we have

$$2^{\alpha(s)} \equiv 1 \pmod{5^s},$$

where $\alpha(s) = \phi(5^s)$, and hence

$$(1) \quad 2^{\alpha(s)} A \equiv A \pmod{2 \cdot 10^s}$$

provided A is divisible by 2^{s+1} . Thus A and $2^{\alpha(s)} A$ have the same s final digits, and their critical digits are of like parity. (We shall refer to the digit preceding the s final digits of any number as its critical digit.)

Now consider the numbers

$$(2) \quad 2^{n+k\alpha(s)}, \quad k = 0, 1, 2, 3, 4.$$

These five numbers have distinct critical digits; for, otherwise we would have

$$2^{n+k\alpha(s)} \equiv 2^{n+k'\alpha(s)} \pmod{10^{s+1}}, \quad 5 > k > k' \geq 0,$$

whence

$$2^{(k-k')\alpha(s)} \equiv 1 \pmod{5^{s+1}};$$

but this is impossible since $(k-k')\alpha(s) < 5\phi(5^s) = \phi(5^{s+1})$, whereas the fact that 2 is a primitive root, mod 5, implies

$$2^r \not\equiv 1 \pmod{5^{s+1}} \quad \text{for } r < \phi(5^{s+1}).$$

This means that if the critical digit of 2^n is odd (or even) then the critical digits of the numbers (2) are, in some order, 1, 3, 5, 7, 9, (or 0, 2, 4, 6, 8). Since by (1) the five numbers (2) have the same s final digits as 2^n , one of them has its $s+1$ final digits exclusively 1's and 2's. This completes the induction.

* No. 4326 by the same proposer: Prove or disprove: 128 is the only power of 2 of two or more digits each of which is a power of 2. (No solution or discussion has been submitted.)

The proof by Starke of Walker's conjecture, as it appeared in the May, 1950 issue of the American Mathematical Monthly.

Again, the conjecture has been proved, and there are infinitely many of each case. Problem B fits the six criteria in the same way as Problem A. Figure 3 again provides a method of solution; namely, to develop successive powers of 2 and search each power for R contiguous zeros. The big difference is that now we will have to retain all of F at every stage, since there is no way to predict where the zeros will appear.

Thus, the two problems are superficially the same, but there are (so far) no shortcuts for Problem B. Figure 8 shows the current status of Problem B, which is now over 20 years old. When last run (on an IBM 7094) it was proceeding at the rate of one new power of 2 every 15 seconds.

Figure 8.

R	X
1	10
2	53
3	242
4	377
5	1491
6	1492
7	6081
8	14007
9	(unknown, but greater than 60,000)

The high precision arithmetic involved in this problem can be done in many ways on a fixed-word-size binary machine. For casual work, the simplest scheme is to operate with one decimal digit per word, which effectively turns the binary machine into a decimal device with a word size (as far as data is concerned) of one digit.

For example, in preparing this article, the values of some of the low order digits of the 89, 189, 289, ... powers of 2 were needed. Figure 9 shows how this was coded quickly and easily for the PeCos machine--but the logic extends to any machine.

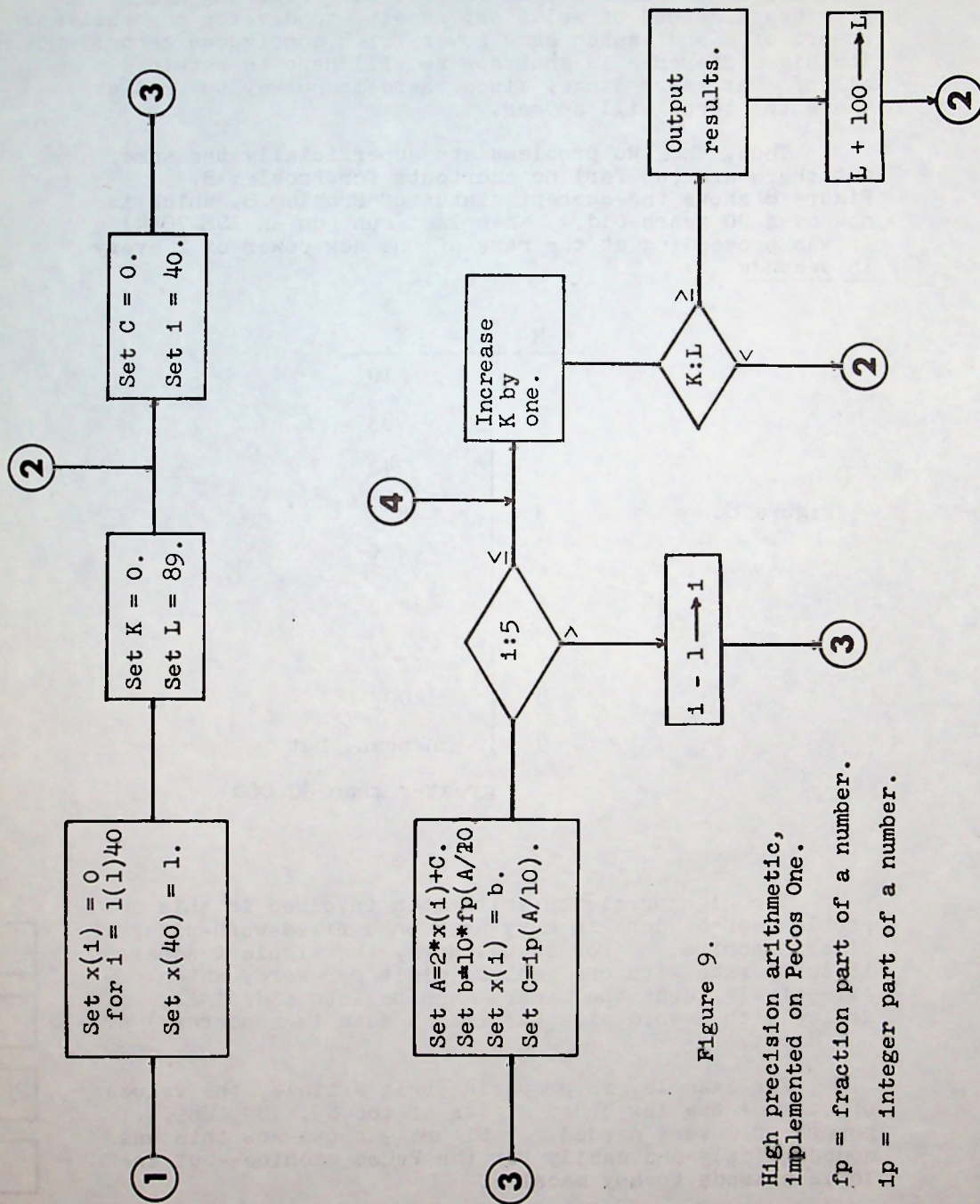


Figure 9.

High precision arithmetic,
implemented on PeCos One.

fp = fraction part of a number.

ip = integer part of a number.

homework

7

PYTHAGOREAN TRIPLETS

We want the logic of a scheme for calculating all the primitive Pythagorean triplets in systematic order.

These are integers that satisfy the Pythagorean relationship:

$$x^2 + y^2 = z^2.$$

The word "primitive" means that the three numbers, X, Y, and Z have no factor in common. Thus, the numbers 5, 12, and 13 form a primitive set, but 10, 24, and 26 do not.

We are assured that if we can find two numbers, R and T, which meet three conditions simultaneously:

- 1) R is greater than T.
- 2) R and T have opposite parity (one odd, the other even).
- 3) R and T have a greatest common divisor of one.

Then X, Y, and Z are given by the equations:

$$X = R^2 - T^2$$

$$Y = 2RT$$

$$Z = R^2 + T^2$$

and these values will form a primitive set.

The problem at hand, then, is that of developing the values of R and T systematically. For each correct pair that is generated, it is proper to state on the flowchart "Calculate and print X, Y, and Z."

It would make little sense to program a check that $x^2 + y^2 = z^2$, since this will be true for any values of R and T. Try R = 4 and T = 6, for example; this pair violates all three conditions, but leads to values of X, Y, and Z that satisfy the Pythagorean relationship.

Arrange to have the values of T build up from low to high within each value of R. Thus, for R = 15, we want to have T take on the values 2, 4, 8, 14, for example. Walk through your final flowchart, to satisfy yourself that after the pair (8,7) for R and T, your logic advances properly to (9,2).

When you have completed your flowchart, re-read this sheet. Is it clear where your flowchart begins? Are all statements used on the flowchart in English? Could someone else follow your logic, or does your flowchart require a personal lecture to be comprehensible?

It is NOT part of this assignment, but plan ahead: how would you test a debugged program that follows the logic of your flowchart? (No program should be committed to production until it is thoroughly tested.) This program is somewhat different from previous ones; this one generates data, so it cannot be tested by furnishing it selected data. Notice that test procedures make excellent exam questions.

Suggestion: As the problem is described above, the 100th set of R and T values will be (22,17), leading to:

$$X = 195$$

$$Y = 748$$

$$Z = 773$$

Does this suggest a possible test procedure?